

CS 312

Internet Concepts

Web Applications: Internet Search and Webpage Accessibility

Dr. Michele Weigle

Department of Computer Science

Old Dominion University

mweigle@cs.odu.edu

<http://www.cs.odu.edu/~mweigle/CS312-F09/>

1

Outline

- ◆ Generic Searching
- ◆ How Google Works
- ◆ Internet Archive
- ◆ Webpage Accessibility
 - » make your webpages more popular
 - » keep your webpages private

2

Generic Search Engines

- ◆ Some very popular ones
 - » Google
 - » Yahoo
 - » MSN

- ◆ Some common characteristics
 - » Huge set of results
 - » Extremely prompt retrieval
 - » Organized retrieval results

How do they achieve such amazing results?

3

How Do Search Engines Work?

- ◆ Prep work
 - » *crawling* – an automated way of browsing the web for pages
 - » archiving, analyzing, organizing, indexing

- ◆ Retrieval
 - » query matching
 - » ranking search results
 - » displaying search results

4

Search Query Syntax

- ◆ The space between keywords is interpreted as AND for some search engines, but as OR for others.
- ◆ Query: I'm interested in cancer in adults.
 - » Boolean logic: AND
 - » Search query: +cancer +adults
- ◆ Query: I'm interested in radiation, but not nuclear.
 - » Boolean logic: NOT
 - » Search query: radiation -nuclear

5

Outline

- ◆ Generic Searching
- ◆ How Google Works
- ◆ Internet Archive
- ◆ Webpage Accessibility
 - » make your webpages more popular
 - » keep your webpages private

6

Google's Three Main Components

- ◆ Crawling
- ◆ Indexing
- ◆ Page Weighting

7

Google's Crawler, the Googlebot

- ◆ Consists of many computers requesting and fetching pages
 - » Googlebot can request thousands of different pages simultaneously
- ◆ When Googlebot fetches a page, it adds all of the links in the page to a queue for subsequent crawling (*deep crawling*)
 - » Googlebot can quickly build a list of links that can cover broad reaches of the web
- ◆ Google continuously recrawls popular frequently changing web pages at a rate roughly proportional to how often the pages change (*fresh crawls*)

8

Submitting URLs to Googlebot

- ◆ <http://www.google.com/addurl.html>
- ◆ Rejects URLs Google suspects are trying to deceive users
 - » including hidden text or links on a page
 - » stuffing a page with irrelevant words
 - » cloaking (aka bait and switch)
 - » using sneaky redirects
 - » creating doorways, domains, or sub-domains with substantially similar content
 - » sending automated queries to Google
 - » linking to bad neighbors.

9

Google's Indexer

- ◆ Parses webpages discovered by the Googlebot
- ◆ Converts them into a set of hits (word occurrences).
 - » For each document in which the word appears, indexer stores the position of the word, font size, capitalization
- ◆ Creates an *inverted index* that is indexed by word pointing to documents

Forward Index

| Document | Words |
|------------|--|
| Document 1 | the, cow, says, moo |
| Document 2 | the, cat, and, the, hat |
| Document 3 | the, dish, ran, away, with, the, spoon |

Inverted Index

| Word | Documents |
|------|--|
| the | Document 1, Document 3, Document 4, Document 5 |
| cow | Document 2, Document 3, Document 4 |
| says | Document 5 |
| moo | Document 7 |

10

Google's Page Weighter, *PageRank*

- ◆ Assigns each webpage a relevancy score
- ◆ A webpage's PageRank depends upon
 - » frequency and location of keywords within the page
 - » how long the webpage has existed
 - » number of other webpages that link to the page
- ◆ Most important factor is number of pages that link to the page
 - » essentially, PageRank is a “vote”, by all the other pages on the Web, about how important a page is

<http://computer.howstuffworks.com/google1.htm>

11

Manipulating PageRank / Webspam

- ◆ Google bomb
 - » is created if many sites link to the page using the same anchor text (increases the page rank)
 - » ex: In 1999, a search for "more evil than Satan himself" resulted in the Microsoft homepage
- ◆ spamdexing
 - » deliberately modifying HTML pages to increase the chance of their being placed close to the beginning of search engine results
- ◆ link doping
 - » embedding a large number of gratuitous hyperlinks on a website, in exchange for reciprocal links
- ◆ Google Jacking (page hijacking)
 - » creating a rogue copy of a popular website which shows contents similar to the original to a web crawler, but redirects web surfers to unrelated or malicious websites.

12

Outline

- ◆ Generic Searching
- ◆ How Google Works
- ◆ Internet Archive
- ◆ Webpage Accessibility
 - » make your webpages more popular
 - » keep your webpages private

13

Recording Internet History

- ◆ Internet Archive
 - » non-profit that was founded to build an Internet library, with the purpose of offering permanent access for researchers, historians, and scholars to historical collections that exist in digital format
 - » <http://www.archive.org>
- ◆ Internet Archive's Wayback Machine
 - » allows you to browse through 85 billion web pages archived from 1996 to a few months ago
 - » <http://www.archive.org/web/web.php>

14

Outline

- ◆ Generic Searching
- ◆ How Google Works
- ◆ Internet Archive
- ◆ Webpage Accessibility
 - » make your webpages more popular
 - » keep your webpages private

15

Web Page Accessibility

- ◆ Are all pages posted on the web necessarily available to all?
- ◆ What about dynamically generated pages?
- ◆ Are there some ways to help enhance accessibility?
- ◆ Is it difficult to be able to restrict access on certain pages?

16

Enhancing Your Page's Popularity

- ◆ Submit your page to search engines
 - » <http://www.google.com/addurl.html>
- ◆ 12 Things to Do to Improve Your PageRank
 - » http://stason.org/articles/money/se/google/12_things_to_do_to_improve_your_site_google_page_rank.html
- ◆ 12 Things **Not** to Do to Improve Your PageRank
 - » http://stason.org/articles/money/se/google/12_things_not_to_do_to_improve_your_site_google_page_rank.html

17

Restricting Accessibility

- ◆ Robots Exclusion Protocol
 - » used to give instructions about a site to web robots (like Googlebot)
 - » instructions are in placed in `/robots.txt` at the root of a website
 - ❖ need to be the website owner to use this
- ◆ Robots can ignore your `/robots.txt`
 - » malware robots - scan the web for security vulnerabilities
 - » email address harvesters used by spammers

<http://www.robotstxt.org/>

18

Examples of /robots.txt

Against indexing the entire site

```
User-agent: *  
Disallow: /
```

*User-agent: **
means all unspecified robots.

Against indexing selected folders or files

```
User-agent: *  
Disallow: /~mweigle/  
Disallow: /~mweigle/files/foo.txt
```

19

Examples of /robots.txt

To allow only specific robots

```
User-agent: googlebot  
Disallow:  
User-agent: yahoobot  
Disallow:  
User-agent: *  
Disallow: /
```

*Does not disallow anything for
googlebot or yahoobot.*

Disallows all to all other robots.

20

What Can a Webpage Author Do?

- ◆ Use robots <meta> HTML tag at the beginning of your webpage
 - » prevent individual pages from indexing

<meta name="robots" content="*tag1, tag2*">

- ◆ Possible *tags*
 - » `noindex` – don't index this page
 - » `nofollow` – don't scan this page for links to crawl
- ◆ Default is `index, follow`

<http://www.robotstxt.org/>

21

Example robots meta tags

Add in the <head> section one of the tags :

```
<meta name="robots" content="index, follow">
<meta name="robots" content="index, nofollow">
<meta name="robots" content="noindex, follow">
<meta name="robots" content="noindex, nofollow">
```

22

Accessibility with Robot Specifications

- ◆ Will all robots follow the instructions?
 - » not the bad guys
- ◆ Can pages still can be accessed when not indexed?
 - » if they're not protected
- ◆ Is there any way available to prevent public access but still allow protected access?
 - » yes!

23

To Prevent Public Access

- ◆ Basic authentication
 - » allows web browsers to provide credentials in the form of a user name and password when making a request
 - » protection is per-directory, not per-file
 - » the requesting client must authenticate itself to the server with a user-ID and a password for each directory being protected
- ◆ Assumes that the connection between the client and server computers is secure and trusted
 - » all information sent in plain-text
- ◆ Could use HTTPS for encryption

24

Basic Authentication and .htaccess

- ◆ Special file named `.htaccess`
 - » must be publicly readable (chmod 644)
 - » '.' is part of the filename
 - ❖ to list files that have a '.', use the 'ls -a' command in Unix
- ◆ States where a corresponding encrypted file (generated in some special way) for this authentication is located
- ◆ Only recommended if you don't have access to main server configuration files
 - » i.e., you don't own the webserver

25

.htaccess Operation

- ◆ When a HTTP server tries to access a directory, it first looks in the directory for this file
- ◆ If this file does not exist at the directory
 - » the server considers that this directory is not protected and accesses the files in it normally.
- ◆ If this file exists at the directory
 - » the server reads this file to find out the location of the encrypted file
- ◆ Server reads the encrypted file and uses the information there to verify authentication

26

.htaccess Operation

- ◆ Before accessing any files in this directory, the server checks the request's authentication header.
- ◆ When the authentication is satisfied by the request information, then the server delivers the requested file in this directory to the requesting client.
- ◆ If not satisfied, the server sends code 401 (unauthorized)
 - » we'll look more at HTTP codes later

27

Example Directory Structure

public_html is the folder where your webpages are located



28

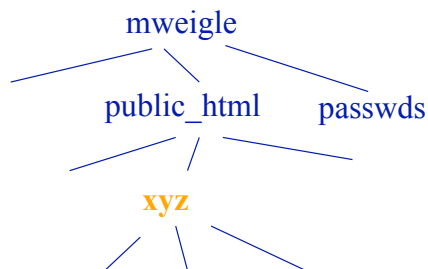
Example .htaccess File

```
AuthType Basic  
AuthName "Password Required"  
AuthUserFile /home/mweigle/passwds/.htpassword
```

29

Example

- ◆ We want to protect all files in the directory
/home/mweigle/public_html/xyz/
- ◆ Only allow user John with password x123y



30

Example

Generate the Encrypted File

- ◆ Run the encryption program `htpasswd` (no dot in the front) as a Unix command
 - » `/usr/apache/bin/htpasswd -c ~/passwds/.htpasswdxyz John`
 - » the parameter `-c` means to create a new encrypted file.
 - » `.htpasswdxyz` is the name of the file to be generated
 - » `John` is the username
- ◆ The program will ask for the password twice

```
% /usr/apache/bin/htpasswd -c ~/passwds/.htpasswdxyz John
New password:
Re-type new password:
Adding password for user John
```

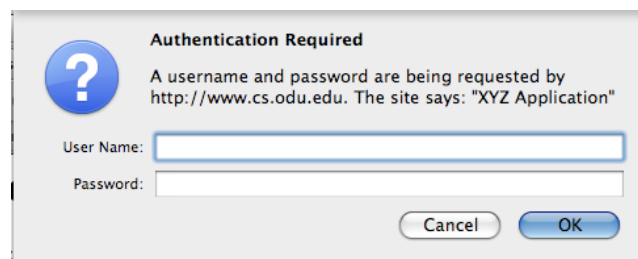
31

Example

Generate the .htaccess File

- ◆ Create a Unix text file named `.htaccess`, stored in `~/public_html/xyz/`

```
AuthType Basic
AuthUserFile /home/mweigle/passwds/.htpasswdxyz
AuthName "XYZ Application"
Require valid-user
```



<http://www.cs.odu.edu/~mweigle/xyz/>

32

Example

File Permissions

- ◆ ~/public_html/xyz/
 - » `chmod 755`
- ◆ ~/public_html/xyz/.htaccess
 - » `chmod 644`
- ◆ ~/passwds/
 - » `chmod 755`
- ◆ ~/passwds/.htpasswdxyz
 - » `chmod 644`

33

More About .htaccess

- ◆ Require
 - » allow any valid user (in the encrypted file)
 - Require valid-user
 - » specify which users can have access
 - Require user John

More info:

<http://httpd.apache.org/docs/2.0/howto/auth.html#basic>

34

Multiple Users

- ◆ Re-run htpasswd once for each new name
- ◆ Make sure not to use `-c` in the command so that the encryptions for the old pairs will not be cleared.
- ◆ The program will again ask for password twice for each new username you give in a command.

35

.htaccess and Access Control

- ◆ Access Control (apart from authentication)
 - » control who can and can't view the page
 - » use 'Allow' and 'Deny' directives
 - Allow from {hostname, IP address, domain name}
 - Deny from {hostname, IP address, domain name, all}
 - » use 'Order' directive to specify the order that they should be applied (usually deny before allow)

| |
|---|
| Order Deny,Allow Deny from all Allow from .cs.odu.edu |
|---|

only allows browsers from .cs.odu.edu domain to view the page or authenticate

Safety of Basic Authentication

- ◆ Only as good as the username and password
- ◆ If not using HTTPS, usernames and passwords are sent in plain-text

37

Outline

- ◆ Generic Searching
- ◆ How Google Works
- ◆ Internet Archive
- ◆ Webpage Accessibility
 - » make your webpages more popular
 - » keep your webpages private

38